REMARKS

Claims 1-28 remain pending in the Application. Claims 1-28 stand rejected by the Examiner. Applicants traverse the rejections of claims 1-28.

Claim Rejections

Claims 1-7, 9-24 and 26-28 stand rejected as being anticipated by "Dynamic Class Loading in the Java™ Virtual Machine" by Liang et al. (hereinafter referred to as Liang). Claims 8 and 25 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Liang in view of U.S. Patent 6,675,381 to Yamaguchi. These rejections are traversed.

Claim 1 provides a method for loading an object class into computer memory for access by applications. Claim 1 recites in combination with its other limitations that a destructible class loader is created such that the destructible class loader is accessed instead of another class loader when a class loading operation is utilized.

For example, the destructible classloader can be made to "look like" a system classloader, such as the system classloader in Java™, and can be released and another new classloader may be added with class changes at anytime (without having to stop and restart the application to install the class changes). The releasing and recreation are performed transparently in that pre-existing classloaders do not need to know that the new "destructible" classloader has been inserted into the class loading hierarchy. The new "destructible" classloader also is not required to know the entire structure of the pre-existing classloading hierarchy since it (and not the system classloader) is automatically returned via the system classloader operation.

Claim 16 recites in combination with its other limitations the modification of a classloader hierarchy such that a different classloader is returned instead of a "normal" or "regular" classloader: "a classloader switching module that inserts a first destructible classloader into the pre-existing class loading hierarchy by causing the class loading operation to provide the first destructible classloader when the class loading operation is called" instead of "one of the classloaders in the pre-existing class loading hierarchy."

Claim 16's dependent claim 18 further illustrates the type of classloader hierarchy modification that may occur. Claim 18 recites that a destructible classloader is returned as the parent of a user classloader instead of a system classloader, and that the system classloader is considered the parent of the destructible classloader.

The Liang reference operates significantly differently from the subject matter recited in such claims. As an illustration, the Liang reference may use a classloader hierarchy, but the reference does not disclose, teach or suggest modifying the classloader hierarchy such that a different classloader is returned instead of the "normal" or pre-existing classloader (e.g., a system classloader). Accordingly, new classloaders as disclosed in the Liang references are not accessed transparently.

[CONTINUED ON THE NEXT PAGE]

For the foregoing reasons, the Liang reference (whether considered alone or in combination with the other cited references) does not disclose, teach, suggest such limitations. Accordingly, Applicant respectfully submit that claims 1-28 are allowable, and the Examiner is respectfully requested to pass this case to issue.

Respectfully submitted,

Date: 5/11/2004

By: _____

John V. Biernacki
Reg. No. 40,511
JONES DAY
North Point
901 Lakeside Avenue
Cleveland, Ohio 44114
(216) 586-3939